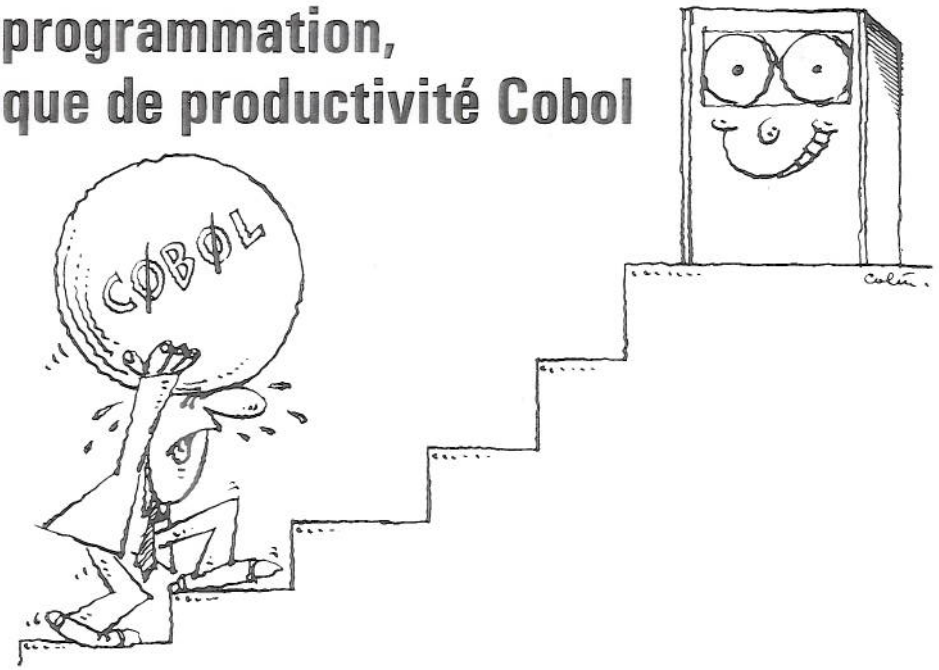
Point de vue

# La préprogrammation, technique de productivité Cobol




LE Cobol, langage le plus répandu actuellement, présente au niveau de son codage des lourdeurs qui ne sont plus compatibles avec la productivité requise d'un personnel opérant sur des systèmes évolués. La rédaction des ordres déjà très lourde est devenue réellement fastidieuse lorsqu'elle nécessite, outre des descriptions d'interfaces, l'utilisation d'instructions non normalisées à multiples paramètres. De leur côté, les générateurs classiques sont longs à mettre en œuvre, difficiles à maintenir et d'un coût prohibitif pour les centres informatiques de tailles moyennes.

Nous pensons qu'il est nécessaire, aujourd'hui, de créer de nouveaux outils « transparents ». Il faut améliorer la productivité des équipes d'études en employant la « préprogrammation » basée sur des utilitaires de type précompilateur. Seul l'usage spécialisé de ces progiciels peut résoudre le problème que pose l'évolution du parc et des possibilités des ordinateurs.

Les nouveaux outils qu'il faut créer doivent fournir un programme source à partir duquel le programmeur travaille. Simples, immédiatement utilisables, ils demandent une connaissance autre que celle requise pour un utilitaire standard. Un langage évolué permet à l'utilisateur de formuler la partie spécifique de son problème ; les programmes résultant sont clairs, structurés, homogènes, aisés à maintenir, portables, optimisés quel que soit le matériel utilisé et le budget mis en œuvre.

Nous estimons que seul un utilitaire de préprogrammation répond à l'ensemble de ces critères.

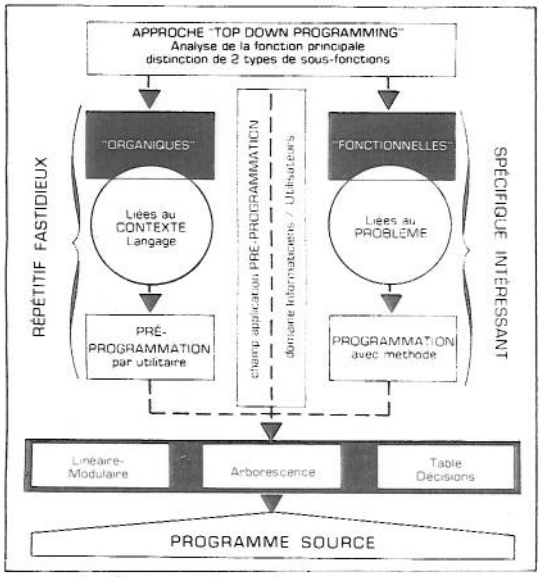
## Un outil pour tous

La programmation passe par la description des données puis leur manipulation de façon élémentaire ou groupée. Un outil modulaire permet de s'adapter à chaque phase de développement.

En mode interactif, un éditeur étudié pour l'emploi du langage Cobol peut servir de noyau et faire appel à des modules spécialisés selon ses besoins. L'un fournit l'« identification » et l'« environnement division », les spécifications des divers fichiers (Select, FD) et les interfaces (Data bases, écran, fichiers spéciaux), un autre se consacre à la description de structures de données en « Data Division » et à l'insertion de groupes d'écritures standard : zones, dates, mois, indices, etc.

La pratique d'une méthode de construction de programme n'interdit pas l'utilisation d'outils d'aides au développement. Les utilitaires sont intéressants à cet égard, donnant d'ailleurs la procédure pour résoudre la partie généralisable des applications. Quant aux parties non généralisables, on peut imaginer l'emploi d'un module acceptant des instructions Cobol classiques et des instructions simplifiées restituées sous leur forme finale. Il n'est pas utile d'aller trop loin dans cette voie, une dizaine de ces dernières instructions étant suffisantes. On peut espérer, de même, des gains de temps importants au niveau des bases de données, de la gestion des écrans et des fichiers spéciaux.

Tous les paragraphes comprenant des instructions de manipulation peuvent être obtenus automatiquement et être appelés par « Perform » aux endroits opportuns des programmes. On peut aussi développer une fonction permettant de décrire un état à partir d'un écran et d'obtenir immédiatement la procédure Cobol correspondante. L'ensemble fournit un programme cohérent, commenté et paginé avec soin, qui est obtenu sans délai, ni erreurs de saisie, qui offre des facilités de maintenance du fait de sa standardisation et de sa portabilité.




La préprogrammation un champ d'application précis.

Le programmeur, comme on le voit, travaille sur le source Cobol, tel qu'il le ferait si ce source avait été obtenu manuellement, mais avec l'avantage d'une base Cobol standardisée. Ce n'est pas de la génération de programme mais de la préprogrammation !

## La préprogrammation

Les échecs en matière de logiciels d'aide à la programmation viennent du caractère empirique de leur développement. Nous pensons qu'il est temps d'acquérir un minimum de maturité en appliquant une méthodologie sérieuse.

L'expérience pratique détermine les besoins, l'étude des problèmes dégage la mise en évidence des concepts théoriques qui les régissent. La technique peut alors suivre et l'outil être développé.

Ces étapes doivent être respectées, mais cela ne suffit pas pour assurer le succès du produit final, le facteur humain intervenant trop souvent.

Ainsi, les théoriciens s'arrêtent à la formulation pure du problème. Mais qui accepte de résoudre la programmation imposée par le « contexte langage » ?

En programmation Cobol, on distingue trois formes principales de structure : linéaire-modulaire, arborescente, table de décision. Ces trois formes découlent toutes d'une approche plus générale — le « Top Down Programming » — consistant à maîtriser parfaitement la structure de la fonction à programmer à l'aide d'un processus d'analyse verticale conduisant à une description en sous-fonctions hiérarchisées. Si l'on rapporte cette conception au contenu d'un programme classique de gestion, on distingue alors deux catégories de natures différentes lorsqu'on affine l'analyse de « forme », qui a conduit au découpage, par une analyse de « fonds » portant sur l'objet de chaque sous-fonction.

Il découle de cette distinction capitale la vision d'un programme scindé en deux parties, l'une « fonctionnelle » correspondant à la formulation spécifique du problème, l'autre « organique » issue des contraintes du langage et de la logique de construction de programme.

Une étude de cette deuxième partie montre qu'elle regroupe toutes les tâches répétitives et fastidieuses de la programmation : c'est le champ d'application de la préprogrammation.

Entre la première et la seconde partie se situe une frontière que la préprogrammation se doit de ne pas franchir, tout regard sur la partie spécifique d'un traitement conduisant obligatoirement à la création d'un codage parallèle au langage évolué, non universel, donc inutile et dangereux. N'oublions pas qu'un outil universel doit être sans ambition méthodologique bien que l'on puisse accepter un certain style de programmation.

## Aide-toi et ta méthode t'aidera

Malgré les nombreuses critiques dont Cobol fait l'objet, nous croyons à son avenir dans la mesure où des méthodes et des outils non improvisés permettront aux programmes d'être considérés comme des investissements à part entière. Cobol est un langage en pleine évolution (la norme 81 le confirme), mais il faut maîtriser cette croissance, quitte à bousculer quelque peu les habitudes des utilisateurs.

Il n'y a pas de miracle à attendre, ni de Cobol, ni des outils d'aide à la programmation, sans effort de méthode. On ne peut parler d'améliorer la productivité et ne pas admettre les outils les plus simples, les moins coûteux, qui pourtant redonneront de l'intérêt au travail des programmeurs. Et au-delà de toute discussion académique, n'y a-t-il pas un fossé, que peu de dirigeants osent franchir, entre la méthode et le risque encouru par son emploi, entre la méthode et la crainte d'un rejet de l'équipe de programmation ?

Il n'y a pas de « voie royale ». La solution passe par une information générale sur le champ d'application de l'outil, préalable à la formation et à son utilisation réelle. Cette utilisation ne doit pas être globale, mais être effectuée d'abord par une mise en œuvre sur les programmes simples avec des personnes intéressées. Ce sera la base d'une future sensibilisation des informaticiens, de façon à créer l'ébauche d'un mouvement irréversible.

Répétons-le : il n'y a pas de voie royale en matière de productivité. Aide-toi et ta méthode d'aidera !

Par contre, si, en théorie, une méthode structurée devrait mener à l'utilisation de produits d'aide à la programmation, en pratique, on observe souvent la démarche inverse : la préprogrammation incite à la structuration par un effet de « feedback » (boucle rétro-entrante). Cette solution est bien pratique lorsque l'on s'attache surtout aux résultats concrets.

**Jean-Pierre Vickoff**